Bounding Buffer Space Requirements for Real-Time Priority-Aware Networks

Hany Kashif and Hiren Patel

Electrical and Computer Engineering
University of Waterloo, Waterloo, Canada
e-mail: {hkashif, hiren.patel}@uwaterloo.ca

Abstract— One implementation alternative for network interconnects in modern chip-multiprocessor systems is priority-aware arbitration networks. To enable the deployment of real-time applications to priority-aware networks, recent research proposes worst-case latency (WCL) analyses for such networks. Buffer space requirements in priority-aware networks, however, are seldom addressed. In this work, we bound the buffer space required for valid WCL analyses and consequently optimize router design for application specifications by computing the required buffer space at each virtual channel in priority-aware routers. In addition to the obvious advantage of bounding buffer space while providing valid WCL bounds, buffer space reduction decreases chip area and saves energy in priority-aware networks. Our experiments show that the proposed buffer space computation reduces the number of unfeasible implementations by 42% compared to an existing buffer space analysis technique. It also reduces the required buffer space in priority-aware routers by up to 79%.

I. Introduction

Real-time embedded applications, and software in general, continue to evolve with increasing complexity. This translates into a need for higher computational power to satisfy their real-time requirements. However, increasing the performance of uniprocessors is not a viable solution anymore due to limits on power consumption and heat dissipation. Though, chip-multiprocessors (CMPs) present a solution to this performance bottleneck, they have only been cautiously adopted as a platform for real-time applications. The reason is that CMPs are designed to optimize average case performance, hence, complicating the provision of real-time guarantees to such applications. As a remedy, recent research focuses on developing custom network-on-chip (NoC) interconnects and their corresponding worst-case latency (WCL) analyses to facilitate their adoption as a platform for real-time applications.

Common NoC implementations include resource reservation (e.g., time-division multiplexing (TDM)) and priority-aware networks. Resource reservation networks statically allocate resources to prevent contention between communication flows in the network at runtime. Æthereal [1] is a NoC that implements TDM. Priority-aware networks, on the other hand, allow contention between communication flows. Contention is resolved at runtime using priority-aware arbitration routers (PARs). TDM networks guarantee schedulability of flows as part of their static allocation of resources and have minimal buffer requirements (one flit for guaranteed services and one packet for best-effort services) [1]. Priority-aware networks have better resource usage compared to TDM but require a worst-case latency (WCL) analysis to guarantee schedulability of the network flows [2, 3].

WCL analysis for real-time applications deployed on priority-aware networks is essential to guarantee honoring their real-time constraints. WCL analysis computes the WCL of communication flows including worst-case interference from other flows. Recent WCL analyses [2, 3, 4, 5] have been developed for priority-aware networks with flit-level preemption. These priority-aware networks employ wormhole switching [6] and virtual channel (VC) resource allocation [7]. While these techniques reduce the required buffer space (flit level) and allow multiple flit buffers (VCs) to access the same physical channel, priority-aware networks are susceptible to chain-blocking (blocked flits spanning multiple routers). Chain-blocking creates back-pressure in the priority-aware network which eventually leads to blocking of the computation and communication tasks. Even though, it is clear that the blocking of tasks due to back-pressure affects the WCL of the communications flows, recent analyses ignore blocking due to back-pressure and, hence, assume infinite buffer space in the network. This assumption is a serious impediment to implementing priority-aware networks because the buffer space required to guarantee the validity of the computed WCLs is unknown.

Multiple works have focused on developing WCL guarantees for priority-aware networks. Shi and Burns [2, 3] present a flow-level analysis (FLA) to compute WCL estimates for the traffic flows. Kashif et al. introduce a link-level analysis (LLA) that provides tighter WCL bounds compared to FLA at the cost of a more detailed analysis. They use LLA for path selection of flows in a priority-aware network [4] and for the computation of end-to-end application WCLs by accounting for offsets between computation and communication tasks in the network [5]. Despite the usefulness of these works in providing WCL guarantees, they never discussed buffer space bounds or inherently assumed the existence of sufficient buffer space. This work, on the other hand, computes the necessary buffer space bounds to ensure the validity of WCLs and the compositionality of holistic WCL estimates.

The main contribution of this work is determining bounds on the buffer space required in the VCs of PARs in the priority-aware network. NoC designs usually target specific applications, and, hence, customizing the design to limit cost, such as buffer space, is applicable [8, 9]. This work computes buffer space bounds to provide an ability to implement priority-aware networks and enable existing WCL analyses to provide timing guarantees for real-time applications. While this paper focuses on computing buffer space bounds, these bounds are usually determined in the chip design phase. However, we can leverage this work for distributing buffer space between the VCs of reconfigurable routers and for guiding mapping and path-

selection algorithms to guarantee a valid WCL analysis. First, we overview priority-aware networks and their respective WCL analyses: FLA and LLA. Then, we present a detailed buffer space analysis using each WCL analysis: namely link-level buffer-space analysis (LLBA) and flow-level buffer-space analysis (FLBA). Our experiments show that, compared to existing buffer space analysis [10], LLBA and FLBA reduce the number of unfeasible implementations by 42% and 27%, respectively. The results also show a reduction in the required buffer space by 79% and 67% using LLBA and FLBA, respectively.

II. Related Work

Since NoC designs usually target specific applications (or application classes), multiple works investigate customizing NoC designs to optimize performance while limiting cost [8, 9]. Some of these works focus on limiting buffer space in NoCs. Hu and Marculescu [11] propose an algorithm for customizing buffer space in NoC routers at a system-level. Given a buffer space budget, they assign buffers to input channels to maximize performance. The proposed work, however, does not consider real-time requirements of flows and does not ensure timeliness. Manolache et al. [10] propose a technique for changing the mapping of data packets to network links and the release timing of packets to reduce buffer sizes and ensure timeliness. While the work does not consider wormhole switching (flit-level preemption) or VC resource allocation, it can be extended to apply to our work. Their proposed approach,however, is orthogonal to the approach presented in this paper. The authors use the worst-case response time analysis (WCRTA) proposed by Palencia and Gonzalez [12] to compute the buffer space requirements. This WCRTA, however, assumes all flows sharing resources (including indirectly interfering flows) to be directly interfering with each other, this produces higher buffer space upper bounds compared to FLBA and LLBA. A similar result applies to WCL bounds as shown in [5]. Kumar et al. [13] present a simulation based algorithm for reducing buffer sizes while considering latency requirements. The authors use simulation to capture the contention between flows on network resources which does not provide worst-case guarantees for real-time flows. Al Faruque and Henkel [14] propose an approach for reducing virtual channel buffers which also does not provide real-time guarantees.

Goosens et al. [1] proposed the Æthereal TDMA-based NoC. They provide a full implementation with buffers for handling best-effort and guaranteed services. Coenen et al. [15] present an algorithm to find the minimal buffer sizes required to decouple computation and communication in the TDMA NoC using credit-based flow control while maintaining real-time guarantees to flows. Similarly, in this work, we attempt to bound the buffer space requirements for priority-aware networks.

III. Background

A. Network Model

In this work, we use the network model proposed in [4, 5] with slight extensions. We assume the deployment of a set of traffic flows to the priority-aware network. Each flow has a source and destination task pair that reside on nodes of the network. The communication between the task pair traverses a set of links in the network which is known as the path of the traffic flow. The priority-aware network employs wormhole switching and a flow-control mechanism to prevent buffer overflow such as the credit flow-control used in the Æthereal NoC [1] and QNoC [9]. Each node in the network consists of a processing element and a PAR.

The set of traffic flows, deployed on the priority-aware network, $\Gamma := \{\tau_i : \forall i \in [1, n]\}$ has n traffic flows. A traffic flow τ_i is a 6-tuple $\langle P_i, T_i, D_i, J_i^R, L_i, F_i \rangle$ with priority P_i, a period T_i, a deadline D_i, and a release jitter J_i^R. We assume that each flow has a distinct priority level. While the proposed approach can be extended to consider priority sharing, we exclude it from this work for brevity and clarity. The flow sends packets either periodically with a period T_i or sporadically with a minimum interarrival time T_i. The basic link latency L_i of a flow is its WCL on one network link when it does not suffer any interferences on that link. The number of flits in one packet of the flow is F_i. The basic link latency can be computed as $\frac{flit\_size * F_i}{bandwidth}$, which is the total packet size divided by the link bandwidth. For clarity of presentation and without loss of generality, we assume that $\frac{flit\_size}{bandwidth} = 1$ to simplify the conversion between latency and buffer space. Note that the proportionality between transmission time and buffer space enables us to cleanly leverage WCL analyses for the buffer space computation.

The path δ_i of a traffic flow τ_i is a sequence of network links $(l_1, \ldots, l_{|\delta_i|})$. The basic latency of flow τ_i over its path is equal to $C_i = L_i + (|\delta_i| - 1)$ where the routing delay is one time unit at each router in the network. A flow is schedulable if its WCL R_i including interferences is less than or equal to its deadline D_i.

B. Router Model

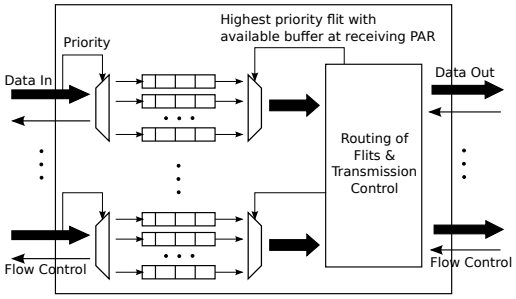


Fig. 1: Priority-aware router architecture [4].

We briefly overview the architecture of the PAR proposed in [2, 3, 4]. Figure 1 shows the internals of the PAR. Each input port has multiple VCs associated with it. A VC is a FIFO buffer that stores flits of a specific priority level. If each flow has a distinct priority level, then a VC will exist for each traffic flow. Otherwise, flows that share the same priority will utilize the same VC. According to a flit's destination and the routing algorithm, the router selects an appropriate output port for the flit. If multiple flits contend for the same output port, the router will forward the flit from the VC with the highest priority. Within the same VC, flits are forwarded based on a FIFO strategy. The flow-control scheme guarantees that a flit is forwarded from one router to another, only when there is enough buffer space at the destination router. If a flit from a certain priority level cannot access the output channel due to insufficient

buffer space at a destination router, then a flit from a lower priority level can access the output channel.

C. Flow-Level Analysis

This section provides an overview of FLA. Additional details on the analysis and the proofs are available in [2, 3]. FLA computes the interference that a flow under analysis τ_i suffers along its path δ_i by considering the whole path as a single shared resource. There are two sources of interference for a packet of the flow under analysis τ_i: (1) direct and indirect interference from higher priority flows and (2) self-blocking from other packets of flow τ_i. Direct interference occurs when a flow τ_j preempts τ_i along its path δ_i. Flow τ_i suffers indirect interference from flow τ_k when flow τ_i has direct interference with flow τ_j which has direct interference with flow τ_k; however, flows τ_i and τ_k do not directly interfere with each other. The set of higher priority flows preempting flow τ_i is denoted by the symbol S_i^D. The set of higher priority flows indirectly interfering with flow τ_i is denoted by the symbol S_i^I.

FLA computes the interference on the path of τ_i to find its worst-case latency. It incorporates indirect interference from flows in the set S_i^I as interference jitter that directly interfering flows suffer. A busy period B_i is the contiguous time interval during which the flow τ_i suffers interference from flows of higher priority and from packets of the same flow. It is given by:

$$B_i = \left\lceil \frac{B_i + J_i^R}{T_i} \right\rceil * C_i + \sum_{\forall \tau_j \in S_i^D} \left\lceil \frac{B_i + J_j^R + J_j^I}{T_j} \right\rceil * C_j$$

The first term computes the latency of the packets of flow τ_i that are released in the busy period B_i. The second term accounts for interference from higher priority flows. The time interval during which interference occurs is $B_i + J_j^R + J_j^I$ where J_j^R and J_j^I are the release and interference jitters of the flow τ_j, respectively. The interference jitter J_j^I is the interference that τ_j suffers from flows in the indirect interference set of τ_i, S_i^I. The number of packets of flow τ_i in the busy period is $p_{B,i}$ which is equal to $\lceil \frac{B_i + J_i^R}{T_i} \rceil$. The worst-case latency R_i of τ_i is then the maximum response time of all packets and is given by:

$$R_i = \max_{p=1 \ldots p_{B,i}} (w_i(p) - (p-1)T_i + J_i^R)$$

where p is the index of the packet of τ_i. This response time of a packet is equal to its worst-case completion time measured from its activation time, $(p-1)T_i$, plus the packet's release jitter. The completion time of job p, $w_i(p)$, in the busy period B_i is given by:

$$w_i(p) = p * C_i + \sum_{\forall \tau_j \in S_i^D} \left\lceil \frac{B_i + J_j^R + J_j^I}{T_j} \right\rceil * C_j$$

D. Link-Level Analysis

LLA performs the WCL analysis at a finer granularity compared to FLA. As a result, LLA computes tighter WCL bounds at the expense of increased computation time. So far, LLA supports both direct and indirect interferences from higher priority flows under the assumption that the deadline of a flow is less or equal to its period [4].

LLA computes the interference on each link along the path of the flow under analysis. It progressively adds new interferences on the flow's path while accounting only once for interferences that are common on consecutive links. If a higher priority flow τ_j interferes with the flow under analysis τ_i on a contiguous set of links, then the interference that τ_j causes should be accounted for only once [5]. The WCL of flow τ_i on any link l along its path is given by:

$$\begin{aligned} \underset{l}{R_i} = \underset{l'}{R_i} &+ \sum_{\forall \tau_j \in \underset{l}{S_i^D}} \left\lceil \frac{\underset{l}{R_i} + J_j^R + J_j^I}{T_j} \right\rceil * L_j \\ &- \sum_{\forall \tau_k \in \underset{l}{S_i^D} \cap \underset{l'}{S_i^D}} \left\lceil \frac{\underset{l'}{R_i} + J_k^R + J_k^I}{T_k} \right\rceil * L_k \end{aligned} \quad (1)$$

where $\underset{l_0}{R_i} = L_i$. The first term represents the WCL on the previous link l'. The second term adds interferences from higher priority flows in the set $\underset{l}{S_i^D}$. The third term subtracts interferences from flows that are common on links l and l', i.e., in the set $\underset{l}{S_i^D} \cap \underset{l'}{S_i^D}$. The WCL on the first link of the path δ_i of flow τ_i is equal to the basic link latency L_i in addition to any interference suffered from higher priority flows on that link. The analysis considers both release and interference jitters as in FLA. For subsequent links, the WCL $\underset{l}{R_i}$ on a link l is equal to the WCL on the previous link l' plus new interferences on link l and subtracting common interferences on links l and l'. The overall WCL of flow τ_i is equal to the WCL on the last link of its path, plus its release jitter, plus the total routing delay along its path. Therefore, the overall WCL R_i is given by:

$$R_i = \underset{l_{|\delta_i|}}{R_i} + J_i^R + (|\delta_i| - 1) \quad (2)$$

IV. Buffer Space Requirements

In this section, we introduce LLBA and FLBA to compute the buffer space required in the PARs of the priority-aware NoC to guarantee a valid WCL analysis. PARs implement a flow-control mechanism to prevent buffer overflow. If a VC in a receiving PAR is full, flits from the corresponding VC in the sending PAR will be blocked until there is space at the receiving PAR buffer. This might lead to chain-blocking and the creation of back-pressure in the network which might invalidate the WCL analysis. Therefore, our goal is to compute the buffer space required for each VC in each PAR to prevent back-pressure in the NoC. For each flow τ_i, we compute the buffer space X_i (measured in flits) required at the VCs used by τ_i along its path δ_i to avoid back-pressure in the network. First, we consider the simplest case:

Lemma 1. *Given a flow τ_i that suffers no interference from higher priority flows ($S_i^D = \emptyset$) and under the condition $D_i \leq T_i - J_i^R$ (no self-blocking), the buffer space required at each VC along δ_i is $X_i = 1$.*

Proof. Since the routers implement the wormhole switching protocol, then each router will directly forward received flits to the next router in the path δ_i of τ_i. For example, if one PAR

forwards a flit to another PAR at time t_1, the receiving PAR will forward it in the next cycle $t_1 + 1$. And since the VCs used by τ_i are used exclusively by τ_i along its path δ_i and no contention occurs for the network links ($S_i^D = \emptyset$), then flits of τ_i are never blocked and are directly forwarded from one router to the other. Therefore, a buffer of one flit size suffices for the VCs of flow τ_i in that case. □

Next, we consider two different interference scenarios:

S 1. *Interference from higher priority flows with no self-blocking from packets of the same flow under analysis.*

S 2. *Interference from higher priority flows with self-blocking from packets of the same flow under analysis.*

We introduce LLBA and FLBA to compute buffer space requirements for each interference scenario along the path of each flow. If enough buffer space exists at each router to accommodate the blocked flits, there will be no back-pressure in the network.

A. Link-Level Buffer-Space Analysis (LLBA)

LLBA computes the buffer space at each PAR on the path of the flow under analysis. Using the interference on a specific link on the path of flow τ_i, we can compute the buffer space required in the VC sending flits of τ_i on that specific link. Theorems 1 and 2 compute the buffer space at each VC for the interference scenarios S 1 and S 2, respectively.

Lemma 2. *Given a flow under analysis τ_i suffering a worst-case interference $\underset{l}{I}_i$ from higher priority flows in the set S_i^D on link l on its path δ_i, an upper bound to the buffer space required at the corresponding VC is given by $\underset{l}{X}_i = \underset{l}{I}_i + 1$.*

Proof. From Lemma 1, the buffer space required at each router along δ_i when no interference exists is one flit. Hence, if $\underset{l}{I}_i = 0$ on link l, a buffer space of one flit will be needed at the corresponding VC, i.e., $\underset{l}{X}_i = 1$. However, if $\underset{l}{I}_i > 0$, more buffer space will be required to prevent back-pressure.

A PAR forwards a flit to each of its output channels every cycle. Consider a flit of τ_i attempting to access link l in a certain cycle. This flit can only be blocked by higher priority flits in the set $\underset{l}{S}_i^D$. In the worst-case, this flit of τ_i will be blocked for a number of cycles equal to $\underset{l}{I}_i$. This will lead to other flits of τ_i accumulating in a FIFO order. To prevent back-pressure, enough buffer space is needed to buffer these flits of τ_i in the same VC. Since a PAR forwards a flit every cycle, then each cycle for which τ_i is blocked causes the accumulation of one more flit in the same VC. Hence, in the worst-case, an extra buffer space of $\underset{l}{I}_i$ flits is need to buffer flits accumulating from a maximum blocking time of $\underset{l}{I}_i$ cycles. Therefore, an upper bound to the buffer space required at the VC buffering flits of flow τ_i to access link l is equal to $\underset{l}{I}_i + 1$. Note that, by definition, the worst-case latency of τ_i is a contiguous time interval during which flits of τ_i and higher priority flits access link l, i.e., including the transmission of flits of flows τ_i. Hence, no interference other than $\underset{l}{I}_i$ can occur before all blocked flits of τ_i can access link l and the VC becomes empty. □

We use Lemma 2 to derive the buffer space requirements for each of the different interference scenarios.

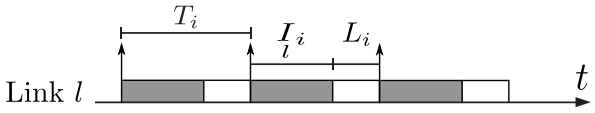


Fig. 2: Interference scenario S 1 on link l.

Theorem 1. *Given a flow under analysis τ_i with WCL $\underset{l}{R}_i$ on link l of path δ_i suffering interference only from higher priority flows in the set $\underset{l}{S}_i^D$ under the condition $D_i \leq T_i - J_i^R$, the buffer space required at the corresponding VC is given by:*

$$\underset{l}{X}_i = \min(L_i, \sum_{\forall \tau_j \in \underset{l}{S}_i^D} \left\lceil \frac{\underset{l}{R}_i + J_j^R + J_j^I}{T_j} \right\rceil * L_j + 1)$$

Proof. Since $\underset{l}{R}_i$ is the WCL of flow τ_i on link l, then the time window during which a higher priority flow τ_j can interrupt τ_i is $R_i + J_j^R + J_j^I$. The number of times that flow τ_j interrupts τ_i is equal to $\lceil \frac{R_i + J_j^R + J_j^I}{T_j} \rceil$. And the interference that τ_i suffers from the flow τ_j is then equal to $\lceil \frac{R_i + J_j^R + J_j^I}{T_j} \rceil * C_j$. The maximum blocking $\underset{l}{I}_i$ that τ_i can suffer from all higher priority flows is equal to the sum of the interference suffered from all higher priority flows. Therefore, using Lemma 2, an upper bound to the buffer space $\underset{l}{X}_i$ is equal to $\sum_{\forall \tau_j \in \underset{l}{S}_i^D} \lceil \frac{R_i + J_j^R + J_j^I}{T_j} \rceil * L_j + 1$.

The required buffer size is, however, also bounded by the number of flits in one packet of the flow, i.e., $F_i = L_i$. We use Figure 2 to prove that both bounds prevent back-pressure in the network. The figure shows flits of flow τ_i (white) accessing link l while suffering interference from higher priority flits (grey). Assume that the first flit of τ_i attempts to access link l at a time t_1 (upward arrow in the figure). This flit will be blocked for an amount of time $\underset{l}{I}_i = \underset{l}{R}_i - L_i$.

Case $\underset{l}{I}_i + 1 < L_i$: At time $t_2 = t_1 + \underset{l}{I}_i$, the buffer will have $\underset{l}{I}_i + 1$ flits and the first flit will be accessing link l, creating an empty slot for a new incoming flit. The last flit in the packet of flow τ_i accesses the link l at time $t_3 = t_1 + \underset{l}{I}_i + L_i$.

Case $L_i < \underset{l}{I}_i + 1$: At time $t_2 = t_1 + L_i$, the buffer will have all flits of the packet under analysis of flow τ_i while the first flit is still blocked. After a blocking time $\underset{l}{I}_i$, the last flit in the buffered packet of flow τ_i will access the link l at time $t_3 = t_1 + \underset{l}{I}_i + L_i$.

In both cases, buffers sizes of $\underset{l}{I}_i + 1$ and L_i flits, respectively, will be sufficient under two conditions: (1) no new interference occurs before time t_3 and (2) no flits of another packet of τ_i arrive to the buffer before time t_3. By definition, the WCL $\underset{l}{R}_i$ is measured from the time the first flit of a packet of τ_i attempts accessing link l (at time t_1) until the time when the last flit of the packet accesses the link (at time t_3). Hence, no interference other than $\underset{l}{I}_i$ (used to compute $\underset{l}{R}_i$) can occur before t_3 or else it

would have been part of $\underset{l}{I}_i$. Thus, satisfying the first condition. The earliest time at which the first flit of a new packet attempts to access link l is $t_4 = t_1 + T_i - J_i^R$. The flit of the new packet will be blocked if $t_4 < t_3$, i.e., $t_1 + T_i - J_i^R < t_1 + L_i + \underset{l}{I}_i$. Since $\underset{l}{I}_i = \underset{l}{R}_i - L_i$, then the blocking occurs when $T_i - J_i^R < \underset{l}{R}_i$. And for schedulability to be satisfied $\underset{l}{R}_i \leq D_i$. So the blocking occurs when $T_i - J_i^R < D_i$ which contradicts the no self-blocking condition $D_i \leq T_i - J_i^R$. Hence, the second condition is satisfied as well. Therefore, $\min(L_i, \underset{l}{I}_i + 1)$ is a safe upper bound to the buffer space X_i to prevent back-pressure. □

To compute the buffer space in the presence of self-blocking, we consider the busy period of flow τ_i on link l. The busy period is the longest contiguous time interval of flits of priority equal to or higher than that of τ_i accessing link l before the link becomes idle.

Theorem 2. *Given a flow under analysis τ_i with a busy period $\underset{l}{B}_i$ suffering interference from higher priority flows in the set $\underset{l}{S}_i^D$ on link l along the path δ_i, the buffer space required at the corresponding VC is given by:*

$$\underset{l}{X}_i = \min(\underset{l}{p}_{B,i} * L_i, \sum_{\forall \tau_j \in \underset{l}{S}_i^D} \left\lceil \frac{\underset{l}{B}_i + J_j^R + J_j^I}{T_j} \right\rceil * L_j + 1)$$

Proof. The proof is similar to the proof of Theorem 1. The busy period $\underset{l}{B}_i$ is the time window during which interference from higher priority flows can occur. Using Lemma 2, an upper bound to the buffer space $\underset{l}{X}_i$ is equal to $\sum_{\forall \tau_j \in \underset{l}{S}_i^D} \lceil \frac{\underset{l}{B}_i + J_j^R + J_j^I}{T_j} \rceil * L_j + 1$. The buffer space is also bounded by the number of flits of τ_i in the busy period which is equal to the number of packets $\underset{l}{p}_{B,i}$ multiplied by the latency (number of flits) in one packet L_i.

If time t_1 is the time at which the first flit of τ_i attempts transmission in the busy period, then the time at which the last flit of τ_i accesses link l is equal to $t_3 = t_1 + \underset{l}{I}_i + \underset{l}{p}_{B,i} * L_i$. We need to prove that the buffer space bound is sufficient under the two conditions mentioned in the proof of Theorem 1. The first condition (no further interference) is satisfied by the definition of a busy period. If further interference occurs before t_3, it would have already been accounted for by $\underset{l}{I}_i$. The first flit of a new packet (beyond the busy period) can attempt to access link l at time $t_4 = t_1 + \underset{l}{p}_{B,i} * T_i - J_i^R$. This new flit can only be blocked if $t_4 < t_3$, i.e., when $\underset{l}{p}_{B,i} * T_i - J_i^R < \underset{l}{I}_i + \underset{l}{p}_{B,i} * L_i$. Since $\underset{l}{I}_i = \underset{l}{B}_i - \underset{l}{p}_{B,i} * L_i$, then blocking happens when $\underset{l}{p}_{B,i} * T_i - J_i^R < \underset{l}{B}_i$. This contradicts the definition of a busy period because if the length of the busy period exceeded the activation time of the new packet, it would have been part of the busy period. Therefore, $\min(\underset{l}{p}_{B,i} * L_i, \underset{l}{I}_i + 1)$ is a safe upper bound to the buffer space X_i to prevent back-pressure. □

B. Flow-Level Buffer-Space Analysis (FLBA)

FLBA is applied for each flow in the network while considering the whole path of the flow as an indivisible unit during analysis. In this case, the buffer space computed using FLA will apply to each router along the path of the flow under analysis. For space constraints, we omit the proofs of the theorems. The theorems are similar to those used by LLBA but applied to the whole path of the flow instead of each link on its path.

Theorem 3. *Given a flow under analysis τ_i with WCL R_i suffering interference only from higher priority flows in the set S_i^D along its path δ_i under the condition $D_i \leq T_i - J_i^R$, the buffer space required at each VC used by τ_i along δ_i is given by:*

$$X_i = \min(L_i, \sum_{\forall \tau_j \in S_i^D} \left\lceil \frac{R_i + J_j^R + J_j^I}{T_j} \right\rceil * C_j + 1)$$

Theorem 4. *Given a flow under analysis τ_i with a busy period B_i suffering interference from higher priority flows in the set S_i^D along its path δ_i, the buffer space required at each VC used by τ_i along δ_i is given by:*

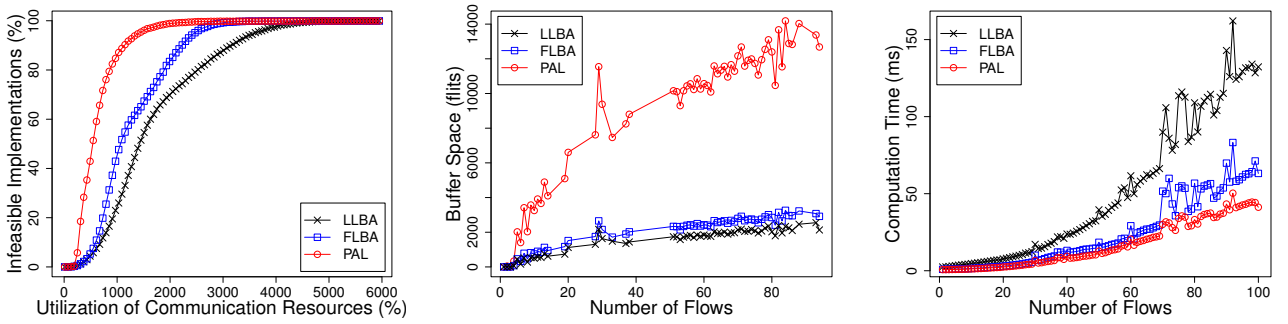
$$X_i = \min(p_{B,i} * L_i, \sum_{\forall \tau_j \in S_i^D} \left\lceil \frac{B_i + J_j^R + J_j^I}{T_j} \right\rceil * C_j + 1)$$

V. Experimentation

We quantitatively evaluate the proposed buffer space computation techniques: LLBA and FLBA. We compare the proposed techniques to the analysis proposed by Palencia and Gonzalez [12] (PAL) which was used for buffer space computation in [10]. We perform the evaluation on a set of synthetic benchmarks as in [10, 4, 5]. Synthetic benchmarks allow us to assess the effect of different factors (as well as their extreme values) on the buffer space computation. We perform our experiments on 4×4 and 8×8 instances of the priority-aware NoC. Our goals from these experiments are: demonstrating the feasibility of computing buffer spaces for priority-aware networks, quantifying the reduction in the number of unfeasible implementations, quantifying the reduction in buffer space bounds computed by the proposed analyses, and comparing the computation times of buffer spaces using PAL, LLBA, and FLBA.

Our experiments involve changing multiple factors to evaluate their affect on buffer space computation. The number of communications flows varies from 1 flow to 100 flows (in steps of 1). The source and destination pairs of the flows are randomly mapped to the NoC. The paths for the flows are computed using a shortest-path algorithm. Packet sizes are uniformly distributed in the range (10,1000) flits. A uniform random distribution is used to assign periods (or minimum interarrival time for sporadic tasks) T_i to communication flows in the range (1000,1000000). The flow's deadline D_i is an integer multiple of its period. An arbitrary priority assignment scheme is used for selecting task priorities. The utilization of the NoC's communication resources varies from 10% to 6000% (in steps of 60%). Hence, we have 40000 possible configurations and we generate 100 different test cases for each configuration.

Figure 3a shows percentage of infeasible implementations against the utilization of the communication resources in the NoC. Increasing the utilization of the network links results in



(a) Percentage of infeasible implementations.

(b) Buffer space against number of flows.

(c) Computation time against number of flows.

Fig. 3: Experimental results.

an increase in the interference suffered by the traffic flows in the network. As the interference increases, some of the flows will require an unbounded (infinite) buffer space. If one flow requires unbounded buffer space, the test case will be considered as an infeasible implementation. The graph shows that for any utilization, LLBA has the least percentage followed by FLBA then PAL. LLBA and FLBA reduce infeasible implementations by 42% and 27%, respectively compared to PAL.

Figure 3b shows the required buffer space against the number of flows at a network utilization of 900%. As the number of flows increases, the required buffer space increases due to the increase of VCs and increase of interference in the network. LLBA has the least buffer space requirements followed by FLBA then PAL On average, LLBA and FLBA reduce the buffer space by 79% and 67%, respectively compared to PAL.

Figure 3c shows the computation time of the buffer space analysis techniques against the number of flows. To guarantee fairness between the different techniques, the computation time includes the time required to run the corresponding WCL analyses needed to apply the buffer space analysis techniques. PAL has the least computation time followed by FLBA then LLBA. On average, LLBA, FLBA, and PAL have computation times of 49 ms, 23 ms, and 16 ms, respectively.

VI. Conclusion

The increase in computational requirements of real-time software and the performance limitations of uniprocessors make CMPs with NoC interconnects a viable platform for real-time software. Although recent research focuses on WCL analysis techniques for priority-aware networks, buffer space requirements were not investigated. Limiting buffer space is also important to reduce silicon area and energy. Typically, NoCs are designed for a specific application or application-classes, hence, designers can customize buffer space based on application requirements. In this work, we extend the two most-recent analyses for WCL computation in priority-aware networks to compute buffer space requirements in PARs which guarantee the validity of the WCL analyses. Our experiments show that LLBA and FLBA reduce the number of unfeasible implementations by 42% and 27% compared to PAL, respectively. LLBA and FLBA also reduce the required buffer space by 79% and 67%, respectively at the expense of an acceptable increase in computation time. Our future work includes developing a complete mapping algorithm for priority-aware networks, using both WCL and buffer space analysis, that guarantees latency bounds and reduces buffer space in the network.

References

[1] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal Network on Chip: Concepts, Architectures, and Implementations," *IEEE Design and Test*, vol. 22(5), 2005.

[2] Z. Shi and A. Burns, "Real-Time Communication Analysis for On-Chip Networks with Wormhole Switching," in *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip*, 2008.

[3] Z. Shi, "Real-Time Communication Services for Networks on Chip," Ph.D. dissertation, The University of York, UK, 2009.

[4] H. Kashif, H. D. Patel, and S. Fischmeister, "Using link-level latency analysis for path selection for real-time communication on nocs," in *Proceedings of the Asia South Pacific Design Automation Conference*, February 2012, pp. 499–504.

[5] H. Kashif, S. Gholamian, R. Pellizzoni, H. D. Patel, and S. Fischmeister, "ORTAP:An Offset-based Response Time Analysis for a Pipelined Communication Resource Model," in *IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2013.

[6] L. Ni and P. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, no. 2, 1993.

[7] W. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, 1992.

[8] L. Benini and G. De Micheli, "Networks on chip: a new paradigm for systems on chip design," in *Proceedings of the conference on Design, Automation and Test in Europe*, 2002.

[9] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *JOURNAL OF SYSTEMS ARCHITECTURE*, vol. 50, 2004.

[10] S. Manolache, P. Eles, and Z. Peng, "Buffer space optimisation with communication synthesis and traffic shaping for nocs," in *Proceedings of the conference on Design, automation and test in Europe*, ser. DATE '06. European Design and Automation Association, 2006.

[11] J. Hu and R. Marculescu, "Application-specific buffer space allocation for networks-on-chip router design," in *IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, 2004.

[12] J. C. Palencia and M. González Harbour, "Schedulability analysis for tasks with static and dynamic offsets," in *Proceedings of the IEEE Real-Time Systems Symposium*, ser. RTSS '98. IEEE Computer Society, 1998.

[13] A. Kumar, M. Kumar, S. Murali, V. Kamakoti, L. Benini, and G. De Micheli, "A simulation based buffer sizing algorithm for network on chips," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2011.

[14] M. A. Al Faruque and J. Henkel, "Minimizing virtual channel buffer for routers in on-chip communication architectures," in *Proceedings of the conference on Design, automation and test in Europe*, ser. DATE '08. ACM, 2008.

[15] M. Coenen, S. Murali, A. Ruadulescu, K. Goossens, and G. De Micheli, "A buffer-sizing algorithm for networks on chip using TDMA and credit-based end-to-end flow control," in *Proceedings of the 4th international conference on Hardware/software codesign and system synthesis*, ser. CODES+ISSS '06. ACM, 2006.